DOGGI-SOB: Discerning Optical Gesture Guidance Integration System On Biped

Tianyi Zhong
Siebel School of Computer Science
University of Illinois
Urbana-Champaign
tzhong8@illinois.edu

Nathaniel Christie
Grainger College of Engineering
University of Illinois
Urbana-Champaign
nmc7@illinois.edu

Andy Mao
Siebel School of Computer Science
University of Illinois
Urbana-Champaign
hanqim2@illinois.edu

Steele Kersten
Siebel School of Computer Science
University of Illinois
Urbana-Champaign
kersten4@illinois.edu

***Abstract*—This project presents DOGGI-SOB, a gesture-controlled bipedal robotic system that integrates real-time computer vision, wireless communication, and embedded locomotion control. The system enables a user to control a dynamically balancing biped robot entirely through monocular hand gestures captured by an onboard camera. A desktop-based Streamlit application receives live video frames over UDP, performs hand landmark extraction using MediaPipe Hands, and classifies gestures into locomotion commands. These commands are transmitted back to the robot through a customized UDP communication interface integrated into the embedded firmware. The firmware continuously interprets incoming commands using an iterative maneuver planner that dynamically updates the robot's motion state in real time. Experimental evaluation characterized gesture recognition robustness, locomotion stability, and end-to-end responsiveness under varying environmental conditions. Results demonstrate that the system can reliably execute forward motion, backward motion, turning, and stop behaviors while maintaining balance and responsive interaction. This work demonstrates the feasibility of intuitive gesture-based control for dynamically balancing robotic platforms using lightweight monocular vision techniques.**

***Index Terms*—Index Terms—gesture recognition, embedded systems, computer vision, bipedal robotics, UDP communication, human-robot interaction.**

I. INTRODUCTION

Human-robot interaction systems increasingly rely on intuitive and natural communication methods to improve usability and accessibility. Traditional robotic control interfaces such as joysticks, keyboards, or remote controllers often require training and direct physical interaction, limiting flexibility in dynamic environments. Hand gesture recognition offers a more natural alternative by allowing users to communicate commands visually without wearable devices or physical contact.

Recent advances in computer vision and lightweight machine learning frameworks have enabled real-time gesture recognition using commodity cameras and embedded hardware. In particular, monocular hand tracking systems such as MediaPipe Hands provide reliable landmark detection with relatively low computational overhead, making them suitable for interactive robotics applications. At the same time, improvements in embedded robotic platforms have enabled dynamically balancing bipeds to perform increasingly complex locomotion tasks.

This project explores the integration of these technologies through DOGGI-SOB (Discerning Optical Gesture Guidance Integration System On Biped), a fully gesture-controlled bipedal robot platform. The goal of the system is to allow a user to intuitively control robot movement using only hand gestures captured through the robot's onboard camera system. The project combines real-time video streaming, gesture recognition, wireless UDP communication, and embedded maneuver planning into a unified control pipeline.

Several challenges arise in such a system. First, real-time gesture recognition requires minimizing latency while maintaining robust landmark detection under varying distances and lighting conditions. Second, wireless communication must reliably transmit commands without introducing stale or delayed control inputs. Finally, the biped controller must safely translate high-level gesture commands into stable locomotion maneuvers without compromising balance.

To address these challenges, we implemented a desktop-based vision interface using Streamlit and OpenCV, integrated MediaPipe-based hand tracking, modified the robot firmware to support continuous UDP gesture commands, and redesigned the maneuver planner into an iterative state-based controller. We further introduced a dynamic pointing mechanism that allows directional turning based on finger position within the camera frame.

Experimental evaluation demonstrates that the integrated system can reliably recognize gestures, execute responsive locomotion commands, and maintain stable operation during continuous user interaction. The results highlight the feasibility of using lightweight monocular gesture recognition as a practical interface for dynamically balancing robotic systems.

II. Deliverables

To first address the goals of the project, the initial goal of testing the viability of the onboard Biped camera was successful. We did not feed to purchase a secondary camera, opting instead to sit in front of the Biped to match its height and camera range.

The following deliverables were promised in the project proposal. They will be described as they were in the project proposal and explained as they are demonstrated in the finished project.

- "A functional gesture recognition program that will be able to, in real time, lock onto a specific person, read that person's body/hand positioning, and interpret that positioning as a specific gesture." This was completed as a hand gesture recognition system, determined to be
- "Several preprogrammed responses to specific gestures, including most importantly moving forward and backward from the user on different gestures and spinning in a circle around the user." The forward and backward responses were implemented but the spinning was abandoned for turning instead.
- "Following the locked in user at a determined distance and stop following separate gestures. / Tracking the locked in user out of camera frame by turning to accommodate their movement." These two stretch goals were implemented via the turning response mentioned earlier. This response took the pointer finger gesture and followed the tip of the finger, turning to face wherever the finger was located.
- "Collision avoidance for preprogrammed responses to prevent the user from making the Biped bump into themselves or the wall." This stretch goal was not implemented, as the Biped only had time of flight sensors on one side of the robot, making a functional collision detection and avoidance system unfeasible.

III. Methodology

Our project integrates computer vision and embedded systems to control a custom bipedal robot entirely via hand gestures. The system architecture is divided into three primary subsystems: a desktop-based video streaming and gesture recognition interface, a customized embedded UDP communication handler, and an iterative, state-based maneuver planner operating on the biped firmware.

A. Video Streaming and Gesture Recognition

The user interface and computer vision pipeline are hosted on a desktop environment using a Python script powered by the Streamlit framework. This application is responsible for receiving real-time JPEG frames from the biped's onboard camera over a UDP network connection.

To ensure the gesture recognition operates on real-time data without latency buildup, we implemented a custom frame overflow mechanism in the socket reading loop. Because UDP sockets buffer incoming packets, processing delays can cause the application to evaluate stale frames. Our mechanism continuously flushes the socket buffer by reading all available packets until a timeout occurs, ensuring that only the most recently received frame is decoded by OpenCV and passed to the neural network.

Once a valid, up-to-date frame is acquired, the script utilizes the MediaPipe Hands solution to extract 21 3D hand landmarks. The Streamlit interface allows users to dynamically adjust detection and tracking confidence thresholds to optimize performance based on lighting conditions. Gestures are classified using heuristic checks on the relative positions of finger joints.

A novel feature of our implementation is the dynamic "pointing" gesture. Rather than assigning a static command to a pointing finger, the script tracks the horizontal (x-coordinate) position of the index finger tip relative to the camera frame. The application divides the visual frame into vertical zones: if the user points towards the left third of the screen, the system issues a "spin_ccw" (counter-clockwise) command; if pointing to the right third, it issues a "spin_cw" (clockwise) command; and if centered, it issues a forward command. This mechanism forces the robot to turn in the direction of the finger until the finger is centered in the camera's field of view. The complete mapping of gestures to biped commands is detailed in Table I.

TABLE I
Mapping of Recognized Hand Gestures to Biped Commands

Recognized Gesture	**Issued Command / Action**
Open Palm	Drive Forward
Peace Sign (V-Shape)	Drive Backward
Pointing (Left Zone)	Spin Counter-Clockwise
Pointing (Right Zone)	Spin Clockwise
Pointing (Center Zone)	Stop
Fist / No Hand	Stop / Hover in Place

B. UDP Communication Modifications

To enable the biped to receive and process the commands generated by the desktop script, we heavily modified the firmware's original UDP message read task (`task.cpp`). We implemented a custom parsing handler that captures the incoming string data (e.g., "forward", "left", "stop"), normalizes it, and maps it to a corresponding integer value. This value is immediately saved to a newly introduced system parameter, `gesture_recognized_`, which is defined in `global.h` to ensure global accessibility across the firmware's RTOS tasks.

A notable trade-off of this approach is the assumption that all incoming UDP traffic on the designated port consists of string-based gesture commands. By overriding the default `BipedMessage` data structure deserialization, our firmware temporarily loses compatibility with the original ground station software. Consequently, the system cannot currently communicate with the ground station, precluding real-time PID parameter tuning or telemetry monitoring while the gesture recognition pipeline is active.

C. Iterative Maneuver Planner

To translate the received network commands into physical robot kinematics, we modified the firmware's original `ManeuverPlanner` class (`maneuver_planner.cpp`). Originally designed to execute a predefined, static sequence of waypoints sequentially, the planner was transformed into a continuous, state-based iterative loop.

During each cycle, the planner evaluates the `gesture_recognized_` global variable and dynamically transitions the biped's controller reference. For example, triggering a `spin_cw` command updates the reference position and target attitude to rotate the biped in place. The planner continues to execute the maneuver corresponding to the latest gesture indefinitely, until a new gesture supersedes it.

To guarantee safety and provide a manual hardware override, we integrated a termination condition using the biped's onboard Button B. We modified the GPIO interrupt service routine associated with this button to trigger a custom termination flag (added to `global.h`). When the interrupt fires, the planner safely halts execution, zeroes the controller references, and exits the planning loop, restoring the biped to a stationary idle state.

IV. Experiments

We evaluated the reliability and safety of the gesture-controlled biped in three stages: (1) gesture recognition characterization, (2) PID and locomotion calibration, and (3) end-to-end integration testing. This approach allowed each subsystem to be validated independently before testing the complete platform.

A. Gesture Recognition Characterization

The first experiments focused on the computer vision subsystem using a USB webcam and the standalone `videostream.py` application. The goal was to determine the practical limits of gesture recognition and refine gesture definitions before deployment on the robot.

We tested recognition performance across varying distances, hand orientations, and lighting conditions. Key metrics included recognition consistency, classification latency, and robustness to partial occlusion. For each gesture in Table I, the operator performed multiple repetitions while varying the hand-to-camera distance from 0.3 m to 2.0 m. We recorded the conditions under which the intended command was correctly identified and noted false positives.

These experiments established several operational constraints. The effective recognition range was approximately 0.5 m to 1.5 m under typical indoor lighting. At shorter distances, the hand often extended beyond the field of view, causing unstable landmark detection. At greater distances, the hand occupied too few pixels for reliable classification. The pointing gesture was particularly sensitive to horizontal placement, motivating the three-zone segmentation strategy described in Section IV-A. Open-palm and no-hand conditions consistently triggered the default stop behavior.

B. PID and Locomotion Calibration

The second stage calibrated the biped's control parameters to ensure that gesture-triggered maneuvers could be executed without loss of balance. Because the robot was originally tuned for predefined trajectories, rapid command changes required adjustments to the proportional-integral-derivative (PID) gains and maneuver velocities.

We tested forward motion, backward motion, and in-place rotation while varying controller gains and target speeds. During each trial, the robot executed a maneuver continuously while operators monitored for excessive tilt and instability. Configurations that caused significant leaning or falls were discarded.

Rotational maneuvers required particular attention because abrupt yaw commands introduced lateral disturbances. The angular velocity and step amplitudes for `spin_cw` and `spin_ccw` were reduced until repeated turns could be completed without falling. Forward and backward speeds were similarly tuned to balance responsiveness and stability.

The final parameters allowed all supported maneuvers to be executed repeatedly without falls under nominal conditions.

C. System Integration and Final Validation

The final stage evaluated the complete system, including wireless video transmission, gesture recognition, UDP communication, and iterative maneuver planning. These tests focused on practical usability and edge cases.

We first tested rapid back-to-back gestures by alternating between commands such as forward, stop, and rotation with minimal delay. The planner consistently adopted the most recent command and transitioned smoothly, confirming that stale UDP packets were not being processed.

We also tested upside-down and rotated gestures. Although MediaPipe continued to detect landmarks, classification accuracy decreased when hand orientation differed significantly from the expected pose.

Recognition was then evaluated while the robot was moving. As the distance between the operator and robot changed, the previously established recognition range of 0.5 m to 1.5 m remained effective.

Finally, we tested the pointing-based turning mechanism. Early versions exhibited overshoot, where the robot continued turning after the finger returned to the center of the frame. This is due to it executing an arcing swing in order to turn. By allowing the biped to change its yaw explicitly rather than changing its position along with its heading, we eliminated this behavior.

The integrated system provided stable and responsive gesture-based control. The robot responded promptly to valid commands, defaulted to a stationary state when no gesture was recognized, and remained balanced during extended operation. These results demonstrate the feasibility of controlling a dynamically balancing biped using monocular hand gesture recognition.

V. CONCLUSION

This project demonstrated the successful integration of computer vision, wireless networking, and embedded robotic control into a fully gesture-controlled bipedal platform. By combining MediaPipe-based hand tracking, real-time UDP communication, and a customized iterative maneuver planner, the system enabled intuitive control of a dynamically balancing robot using only monocular hand gestures.

Experimental evaluation showed that the platform could reliably recognize gestures within practical operating distances while maintaining stable locomotion during forward movement, backward movement, and rotational maneuvers. The introduction of socket buffer flushing and continuous planner updates reduced latency and prevented stale commands from affecting robot behavior. In addition, the dynamic pointing gesture provided a more natural turning mechanism by allowing the robot to align itself with the user's indicated direction.

Despite the successful integration, several limitations remain. Gesture recognition performance degrades under extreme lighting conditions, large hand rotations, and increased distances from the camera. Furthermore, the modified UDP communication pipeline sacrifices compatibility with the original ground station software during operation. Future work could address these limitations by incorporating more robust machine learning-based gesture classifiers, bidirectional communication protocols, and autonomous tracking capabilities. Additional improvements may include onboard inference directly on embedded hardware to reduce network dependency and latency.

Overall, this work demonstrates that lightweight monocular vision techniques can provide an effective and intuitive interface for controlling dynamically balancing robotic systems in real time.